Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни

«Алгоритми та структури даних»

Виконав: студент групи ІМ-42

Максим Крамаренко Юрійович

номер варіанту: 17

Перевірив:

Сергієнко А. М.

Київ 2025

Постановка задачі

1. Представити у програмі напрямлений і ненапрямлений графи з заданими параметрами:
   • кількість вершин n;
   • розміщення вершин;
   • матриця суміжності A.
2. Створити програму для формування зображення напрямленого і ненапрямленого графів у графічному вікні. Згадані вище параметри графа задаються на основі чотиризначного номера варіанту n1n2n3n4, де n1n2 це десяткові цифри номера групи, а n3n4—десяткові цифри номера варіанту, який був у студента для двох попередніх робіт (див. таблицю з поточними оцінками з АСД, надану викладачем на початку поточного семестру). Кількість вершин nдорівнює 10 n3. Розміщення вершин:
   • колом з вершиною в центрі при n4 = 6,7;

3. Матриця суміжності Adir напрямленого графа за варіантом формується таким чином:
   1) встановлюється параметр (seed) генератора випадкових чисел, рівний
   номеру варіанту n1n2n3n4
   2) матриця розміром n n заповнюється згенерованими випадковими чи-
   слами в діапазоні [0, 2.0);
   3) обчислюється коефіцієнт k = 1.0 – n3*0.02 – n4*0.005 – 0.25;
   4) кожен елемент матриці множиться на коефіцієнт k;

5) елементи матриці округлюються: 0 — якщо елемент менший за 1.0, 1 — якщо елемент більший або дорівнює 1.0.
Матриця суміжності Aundir ненапрямленого графа одержується з матриці Adir

Варіант 17:

n1 = 4
n2 = 2
n3 = 1
n4 = 7

n = n3 + 10

k = 1.0 - n3 * 0.02 - n4 * 0.005 - 0.25

random.seed(4217)

Текст програми:

```
import random
import math

n1 = 4
n2 = 2
n3 = 1
n4 = 7

n = n3 + 10

k = 1.0 - n3 * 0.02 - n4 * 0.005 - 0.25

random.seed(4217)

print("Directed graph:")
dir = [[0 for _ in range(n)] for _ in range(n)]
```

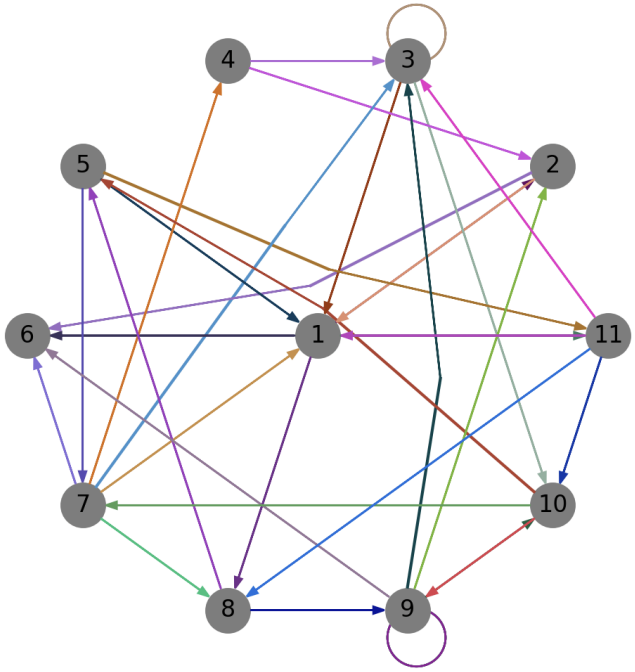
```
for i in range(n):
    for j in range(n):
        dir[i][j] = math.floor(random.uniform(0, 2.0) * k)
        print(dir[i][j], end = ' ')
    print()

print("\n\n")

print("Undirected graph:")
undir = [[0 for _ in range(n)] for _ in range(n)]
for i in range(n):
    for j in range(n):
        undir[i][j] = max(dir[i][j], dir[j][i])
        print(undir[i][j], end = ' ')
    print()

import matplotlib.pyplot as plt

# Function to draw the graph
def draw_graph(matrix, directed=False):
    R = 10  # R of the circular layout
    angle_step = 2 * math.pi / (n - 1)  # Angle between nodes

    # Calculate node positions
    positions = []
    for i in range(n):
        if i == 0:
            positions.append((0, 0))
            continue
        angle = i * angle_step
        x = R * math.cos(angle)
        y = R * math.sin(angle)
        positions.append((x, y))

    node_R = 0.8

    # Plot the nodes
    plt.figure(figsize=(8, 8))
    for i, (x, y) in enumerate(positions):
        plt.scatter(x, y, s=500, color="gray", zorder=2)  # Draw node
```

```
        plt.text(x, y, str(i + 1), fontsize=12, ha="center",
va="center", zorder=4)  # Label node

    # Helper function to adjust for node boundary
    def adjust_for_R(x1, y1, x2, y2, offset):
        dx, dy = x2 - x1, y2 - y1
        length = math.sqrt(dx ** 2 + dy ** 2)
        if length == 0:
            return x1, y1, x2, y2
        scale = (length - offset) / length
        return x1 + dx * (1 - scale), y1 + dy * (1 - scale), x2 - dx *
(1 - scale), y2 - dy * (1 - scale)

    # Plot the edges
    m = 1
    def rand(a):
        return random.choice([random.uniform(-2*a,(node_R + a/8)),
random.uniform((node_R + a/8),2*a)])

    for i in range(n):
        for j in range(n):
            if matrix[i][j]:
                # Draw self-loop if a node connects to itself
                if matrix[i][i]:
                    x, y = positions[i]
                    loop_radius = 1  # Adjust for better visibility
                    vector_length = math.sqrt(x ** 2 + y ** 2)
                    x += x * loop_radius / vector_length
                    y += y * loop_radius / vector_length
                    loop = plt.Circle((x, y), loop_radius,
color=edge_color, fill=False, zorder=1)
                    plt.gca().add_patch(loop)
                x1, y1 = positions[i]
                x2, y2 = positions[j]

                dx, dy = x2 - x1, y2 - y1
                length = math.sqrt(dx ** 2 + dy ** 2)
                x1, y1, x2, y2 = adjust_for_R(x1, y1, x2, y2, node_R)
```

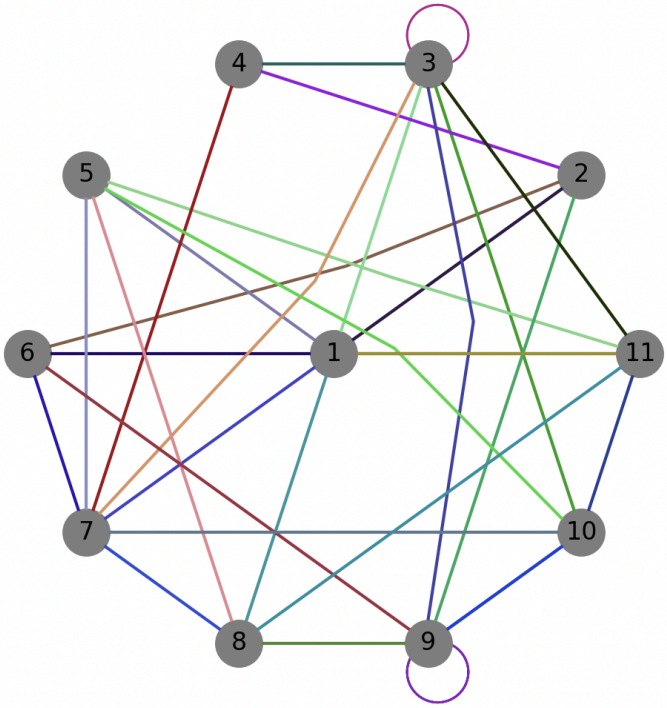
```
                edge_color = (random.randint(0, 235) / 255,
random.randint(0, 235) / 255, random.randint(0, 235) / 255)  # Generate
a random RGB color
                if (length >= 2*(R - node_R) and length <= 2*(R +
node_R)):# If the line goes through the center
                    midx = (x1 + x2) / 2 + rand(m)
                    midy = (y1 + y2) / 2 + rand(m)
                    # Draw directed edge (arrow)
                    if directed:
                        plt.plot([x1, midx], [y1, midy],
color=edge_color, zorder=3)
                        plt.arrow(midx, midy, x2 - midx, y2 - midy,
head_width=0.25, length_includes_head=True, color=edge_color, zorder=4)
                        continue
                    plt.plot([x1, midx, x2], [y1, midy, y2],
color=edge_color, zorder=3)
                    continue
                if directed:
                    plt.arrow(x1, y1, x2 - x1, y2 - y1, head_width=0.25,
length_includes_head=True, color=edge_color, zorder=4)
                    continue
                plt.plot([x1, x2], [y1, y2], color=edge_color, zorder=3)

    # Set plot limits and hide axes
    plt.xlim(-15, 15)
    plt.ylim(-15, 15)
    plt.axis("on")
    plt.show()

# Draw the directed graph
draw_graph(dir, directed=True)

# Draw the undirected graph
draw_graph(dir, directed=False)
```

Тестування програми:

```
Directed graph:
0 1 0 0 0 1 0 1 0 0 1
1 0 0 0 0 1 0 0 0 0 0
1 0 1 0 0 0 0 0 0 1 0
0 1 1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 0 1 0 1 0 0 0
0 0 0 0 1 0 0 0 1 0 0
0 1 1 0 0 1 0 0 1 1 0
0 0 0 0 1 0 1 0 1 0 0
1 0 1 0 0 0 0 1 0 1 0
```




```
Undirected graph:
0 1 1 0 1 1 1 1 0 0 1
1 0 0 1 0 1 0 0 1 0 0
1 0 1 1 0 0 1 0 1 1 1
0 1 1 0 0 0 1 0 0 0 0
1 0 0 0 0 0 1 1 0 1 1
1 1 0 0 0 0 1 0 1 0 0
1 0 1 1 1 1 0 1 0 1 0
1 0 0 0 1 0 1 0 1 0 1
0 1 1 0 0 1 0 1 1 1 0
0 0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 0 1 0 1 0
```



Висновок:

Засвоїв теоретичний матеріал лекцій набрався практичного досвіду в створенні та відображені напрямлених і ненапрямених графів. Навчився працювати з графічними вікнами та функціями для них. Збільшив досвід роботи з матрицями.